**Problem Statement**

Explain the different methods of

● mapper class

● reducer class in brief.

**Solution**

**#Various Methods of mapper Class**

There are 4 methods of Mapper class

1.Setup

2.Map

3.run

4.Cleanup

**#SETUP method**

It is called once at the start of the task

Uses:

It is used for intialising some common values for all Mappers.For example while using MapSide join we we will use setup method to add the files from distributed cache to all the nodes where mapper is running by setup method

Example

@Override  
protected void setup(Context context) throws IOException,InterruptedException {

Path[] cacheFilesLocal = DistributedCache.getLocalCacheFiles(context.getConfiguration());

for (Path eachPath : cacheFilesLocal) {  
if (eachPath.getName().toString().trim().equals(“departments\_txt”)) {  
context.getCounter(MYCOUNTER.FILE\_EXISTS).increment(1);  
loadDepartmentsHashMap(eachPath, context);  
}

**#Map Method**

It is called one key value pair in the input split since record reader by default takes line start byte offset as key and line as value map method is invoked for each line in the file

Example

Example:

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString());

while (itr.hasMoreTokens()) {

word.set(itr.nextToken());

context.write(word, one);

}

}

}

**#Run Method**

Expert users can override this method for more complete control over the execution of the Mapper.

the MR framework actually calls the run() which will the call the setup(), map() n times and finally cleanup()

**cleanup method**

It is called once at the end of the task

It can be used for writing the map output on context after all mapper gets over.

**#Different Methods Of Reducer Class**

1.Setup

2.Reduce

3.run

4.Cleanup

**#SETUP method**

It is called once at the start of the task.It is used for setting some common values to all reducers

**#Reduce Method**

**This method is called once for each unique key**

**Example**

public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text, IntWritable>

{

public void reduce(Text word, Iterable<IntWritable> values, Context con) throws IOException, InterruptedException

{

int sum = 0;

for(IntWritable value : values)

{

sum += value.get();

}

con.write(word, new IntWritable(sum));

}

**#Run method**

Advanced application writers can use the [run(org.apache.hadoop.mapreduce.Reducer.Context)](https://hadoop.apache.org/docs/r2.6.2/api/org/apache/hadoop/mapreduce/Reducer.html#run(org.apache.hadoop.mapreduce.Reducer.Context)) method to control how the reduce task works.

**#Cleaup Method**

Called once at the end of the Task

It is used for finally writing in context as if we write in reduce it will be written for each key

Example

Protected void cleanup(Context context) throws IOException, InterruptedException {

Map sortedMap = sortByValues(countMap);

int counter = 0;

for (Text key: sortedMap.keySet()) {

if (counter ++ == 20) {

break;

}

context.write(key, sortedMap.get(key));

}